BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

PCM ok

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 5 : G06F 9/445

A1

(11) International Publication Number: WO 94/11812

(43) International Publication Date: 26 May 1994 (26.05.94)

(21) International Application Number: PCT/US93/11227

(22) International Filing Date: 16 November 1993 (16.11.93)

(30) Priority data:
07/976,945 16 November 1992 (16.11.92) US

(71) Applicant: MICROSOFT CORPORATION [US/US]; One Microsoft Way, Redmond, WA 98052-6399 (US).

(72) Inventor: SPIES, Terence, Ronald ; 4850 - 156th N.E., Apartment 308, Redmond, WA 98052 (US).

(74) Agents: RONDEAU, George, C., Jr. et al.; Seed and Berry, 6300 Columbia Center, 701 Fifth Avenue, Seattle, WA 98104-7092 (US).

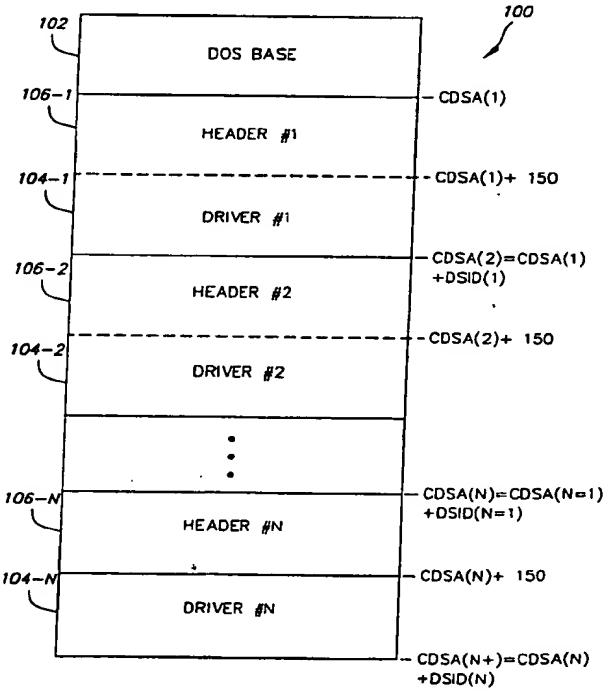
(81) Designated States: CA, JP, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: METHOD FOR LOADING DEVICE DRIVERS

(57) Abstract

A method for storing device drivers in the adapter region of a computer's address space includes the step of providing a plurality of headers and associating each header with a device driver to be loaded and initialized. Each header includes instructions for loading and initializing its associated device driver. Further, each header includes information for identifying the address next following its device driver. The method further includes the step of storing the plurality of device drivers and their corresponding headers in an address space having addresses that are sequential. Further, each header is stored in an address space having addresses that are sequential with the addresses of the address space occupied by its corresponding device driver. A method is also disclosed for loading the plurality of device drivers stored in the manner discussed above. The method for loading comprises the step of locating a header and executing the instructions for loading and initializing its corresponding device driver. Thereafter, the driver size identifier is used to identify the location of the header next following the device driver just loaded.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria	GB	United Kingdom	MR	Mauritania
AU	Australia	GE	Georgia	MW	Malawi
BB	Barbados	GN	Guinea	NE	Niger
BE	Belgium	GR	Greece	NL	Netherlands
BF	Burkina Faso	HU	Hungary	NO	Norway
BG	Bulgaria	IE	Ireland	NZ	New Zealand
BJ	Benin	IT	Italy	PL	Poland
BR	Brazil	JP	Japan	PT	Portugal
BY	Belarus	KE	Kenya	RO	Romania
CA	Canada	KG	Kyrgystan	RU	Russian Federation
CF	Central African Republic	KP	Democratic People's Republic of Korea	SD	Sudan
CG	Congo			SE	Sweden
CH	Switzerland	KR	Republic of Korea	SI	Slovenia
CI	Côte d'Ivoire	KZ	Kazakhstan	SK	Slovakia
CM	Cameroon	LI	Liechtenstein	SN	Senegal
CN	China	LK	Sri Lanka	TD	Chad
CS	Czechoslovakia	LU	Luxembourg	TG	Togo
CZ	Czech Republic	LV	Latvia	TJ	Tajikistan
DE	Germany	MC	Monaco	TT	Trinidad and Tobago
DK	Denmark	MD	Republic of Moldova	UA	Ukraine
ES	Spain	MG	Madagascar	US	United States of America
FI	Finland	ML	Mali	UZ	Uzbekistan
FR	France	MN	Mongolia	VN	Viet Nam
GA	Gabon				

Description

METHOD FOR LOADING DEVICE DRIVERS

Technical Field

The present invention is directed toward a method for loading device drivers and, more particularly, a method for loading device drivers without changing the underlying drivers, without using a disc memory, and with minimum overhead.

Background of the Invention

Personal computers are becoming more and more commonplace in today's society. Personal computers typically include a central processing unit (CPU) that is controlled by basic instructions referred to as the operating system of the computer. Generally, the personal computer requires one or more peripheral devices to extend its capabilities. Typical peripheral devices include keyboards, display screens, printers, disc drives, modems, *etc.* Typically, a device driver is required to interface the operating system of the personal computer with the peripheral devices.

Device drivers are small computer programs that receive instructions and data from the operating system that are independent of the peripheral device for which the instructions and data are intended. The device driver responds to the device independent instructions and data from the operating system to provide instructions and data to the peripheral device that are dependent upon the nature and construction of the particular peripheral device. Accordingly, device drivers will typically provide additional information to the peripheral device to enable the peripheral device to perform the functions requested by the operating system. By providing different device drivers with different peripheral devices, the operating system of the computer can interface with peripheral devices made from different manufacturers.

So that the personal computer can interface with all of its peripheral devices, the device drivers must be loaded into the computer memory and initialized prior to operation of the personal computer. Typically, loading the device drivers usually occurs when the personal computer is turned on. When the personal computer is turned on, the operating system performs functions necessary prior to operation. As mentioned above, one of these

functions is to load the device drivers from a permanent storage location into the computer memory.

Presently, two methods exist for loading and initializing device drivers into the computer memory when the computer is turned on. The first method relies upon configuration data, a file typically called CONFIG.SYS., being stored in the computer's memory to determine which device drivers are stored in permanent memory and to load and initialize these device drivers. When a new device, and device driver, are added to the personal computer, the CONFIG.SYS file is updated to include information identifying the new driver, its location on the disk, and the location of its initialization program. Accordingly, when the CONFIG.SYS file is accessed, it includes all necessary information to locate each device driver and the routine for loading and initializing the device driver. The CONFIG.SYS file then executes these routines so that each device driver is loaded and initialized.

The other presently available method for loading and initializing device drivers is referred to as ROMDOS. The ROMDOS method enables the device drivers to be stored in read only memory (ROM). To load the device drivers, however, the ROMDOS method relies on the CONFIG.SYS file. Accordingly, the ROMDOS method must provide instructions and data to simulate a hard disk so that the device drivers stored in read only memory (ROM) can be accessed using the CONFIG.SYS file. Further, the ROMDOS method relies upon a subprogram, typically referred to as CONFPROC.SYS, to interface with the CONFIG.SYS file thereby enabling the CONFIG.SYS file to identify and load the device driver stored in read only memory. Although effective, this method requires unnecessary memory for storing the CONFPROC.SYS subprogram. Further, this method creates additional inefficiencies in operating the computer and in storing the device drivers.

Accordingly, it is desirable to provide a method for storing device drivers in read only memory so that the device drivers can be loaded with minimum storage overhead and without changing the underlying device drivers.

Summary of the Invention

The present invention is a method for storing a plurality of device drivers in the adapter region of an address space of a computer so that the device drivers can be loaded into a computer memory of the computer. The method includes the steps of providing a header for each device driver wherein each header includes instructions and data for loading its respective device

driver and wherein each header includes a driver size identifier. The method further includes the step of storing the plurality of device drivers and headers in a portion of the adapter region so that each header occupies an address space with addresses that are in sequence with the addresses of the address space occupied by its respective device driver. The plurality of device drivers and headers are further stored so that they occupy an address space having addresses that are sequential. The driver size identifier of each header is provided to identify the number of addresses of the address space occupied by the header and its respective device driver. The plurality of device drivers and headers are further stored so that the first address of a subject header and its driver size identifier can be combined to provide the first address of another header next following the subject header.

In operation, the method includes the steps of identifying a current driver scan address. The method also includes the step of accessing the current driver scan address and determining whether the information stored at the current driver scan address is associated with a header and, if so, performing the remaining steps, and, if not, determining that all drivers have been loaded. After the current driver scan address has been accessed, the instruction of the header associated with the current driver scan address are executed to load the device driver associated with the header. Thereafter, a driver size identifier of the header is accessed and combined with the current driver scan address to identify the new current driver scan address. The method is repeated until the determination is made that all drivers have been loaded.

Brief Description of the Drawings

Figure 1 is a diagram illustrating the data structure for storing device drivers in the adapter region of a computer's address space;

Figure 2 is a diagram illustrating the data structure for a header to be associated with a device driver as shown in Figure 1; and

Figure 3 is a decision flow diagram illustrating the method for loading device drivers from the adapter region of a computer's address space.

Detailed Description of the Invention

The present invention is a method for storing device drivers in the adapter region of a computer's address space. Further, the subject invention includes the method for loading the device driver into the computer's memory from the adapter region.

As is known in the art, personal computers are typically provided with a central processing unit (CPU) that defines an address space of the computer. The address space is typically divided into an adapter region, sometimes referred to as an upper memory region, and computer memory. Commonly, the computer memory refers to random access memory (RAM) that is used by the computer for storing programs, instructions, data, and other information with which the central processing unit is currently working. The adapter region is commonly used for positioning memory and other devices to be accessed by the central processing unit at times when it is not accessing its computer memory.

A particularly unique feature of the present invention is the ability to store device drivers in the adapter region of the computer's address space. The device drivers are stored in a manner so that the amount of memory necessary to store additional information for loading and initializing the device driver is minimized. A data structure 100 illustrating the manner in which the device drivers are stored in the adapter region is shown in Figure 1.

The data structure 100 includes an operating system portion 102, referred to in Figure 1 as DOS Base. The operating system portion 102 is a portion of the operating system of the computer that is stored in the adapter region for facilitating the identification, loading, and initialization of the device drivers, as will be described more fully below. The operating system portion 102 is preferably stored at a predetermined address of the adapter region so that it can be readily identified by the remaining portion of the computer's operating system. However, it will be apparent to those skilled in the art that many techniques can be employed to identify the operating system portion 102.

Generally, the operating system portion 102 is provided to identify the location of the device drivers to be loaded into the computer's memory. For this reason, the operating system portion 102 illustrated in Figure 1 includes an operating system size identifier for identifying the size of the operating system portion. Preferably, the plurality of device drivers will immediately follow the operating system portion 102, as will be discussed below. Therefore, by identifying the size of the operating system portion 102, *i.e.*, the number of addresses associated with the address space occupied by the operating system portion, then the first address of the operating system portion can be mathematically combined with the operating system size identifier to provide the identity of the first address wherein the device driver information will be found.

The operating system portion 102 further includes a common driver signature that is used to identify the last device driver, as will be discussed more fully below.

The data structure 100 further includes a plurality of device drivers 104-1 through 104-N, where N is the total number of device drivers stored. In accordance with the subject invention, a plurality of headers 106-1 through 106-N are each associated with a respective device driver. As shown in Figure 2 with respect to one header 106-i, the plurality of headers include a driver signature 200 (Figure 2), a driver size identifier (DSID) 202, and instructions and data 204 for loading and initializing its associated device driver 104-i (not shown).

In accordance with the presently preferred embodiment of the invention, the common driver signature 200 that is stored in the operating system portion 102, as discussed above, is also stored at the first address of the header 106-i, the first address being referred to herein as the current driver scan address (CDSA). Further, the common driver signature 200 is a 2 byte signature followed by the driver size identifier (DSID) 202. The driver size identifier 202 occupies another 2 bytes of memory and is followed by the instructions and data 204.

An important aspect of the subject invention is that the operating system portion 102 is able to locate each header. Accordingly, by knowing the current driver scan address (CDSA) of any header 106-i, the computer can access the common driver signature 200, the driver size identifier (DSID) 202, or the instructions and data 204 for loading and initializing the associated device driver. Still further, by knowing the current driver scan address (CDSA), the computer also knows the first address of the associated driver 104-i.

Returning to Figure 1, in accordance with the present invention, the plurality of device drivers and headers are each stored in a portion of the adapter region directly following the operating system portion 102. With reference to Figure 1, it will be understood that the address space occupied by the operating system portion 102 has addresses that are lower than the addresses occupied by the header 106-1. Similarly, the addresses of the address space occupied by the header 106-1 are lower than the addresses of the address space occupied by the driver 104-1, *etc.* It is significant, however, that the addresses of the address space occupied by the drivers 104-1 through 104-N and the headers 106-1 through 106-N are sequential. It will be further appreciated that the header 106-1 associated with the driver 104-1 occupies an address space

having addresses that are sequential with the addresses of the address space occupied by its associated driver 104-1. In this manner, the driver size identifier referred to in Figure 2 can be combined with the current driver scan address and the predetermined size of the header to determine the location of the next header 106-i and its associated driver 104-i to be loaded. In this manner, the plurality of drivers can be readily identified, loaded, and initialized using the headers 106-i.

Referring to Figure 3, a decision flow diagram is provided illustrating the method for identifying the device drivers to be loaded into the computer memory. Initially, the operating system of the computer will locate and access the operating system portion 102 (DOS Base) and determine a current driver scan address (CDSA), step 300. As discussed above, the operating system portion 102 includes an operating system size identifier for determining the first address following the operating system portion 102. That address is identified as the current driver scan address (CDSA).

After determining a current driver scan address (CDSA), the computer determines whether the information located at the current driver scan address (CDSA) is the common driver signature 200 (Figure 2) and, if not, determines that all device drivers have been loaded, step 302. If, however, the information located at the current driver scan address (CDSA) is the common driver signature, then the computer will execute the instructions and data 204 of the header 106-i to load and initialize the device driver 104-i, step 304. After the device driver has been loaded and initialized, the computer will locate the driver size identifier (DSID) 202 of the header 106-i, step 306. The driver size identifier (DSID) 202 is then combined with the current driver scan address (CDSA) to determine the new current driver scan address, step 308. Thereafter, step 302 is repeated to determine whether the new current driver scan address includes a driver signature.

It will be apparent to those skilled in the art that, when the driver signature is found, the computer knows it has located still another header and, therefore, determines that another device driver remains to be loaded and initialized. Further, since the plurality of headers and device drivers are stored in a portion of the address space of the adapter region having sequential addresses, if the new current driver scan address (CDSA) does not include the driver signature, then all device drivers have been loaded.

Still further, significant advantages can be realized from the method of the subject invention. Generally, in accordance with the above-

described method of the subject invention, the operating system will provide the header 106-i with a pointer to an address where the device driver is to be loaded. During execution of step 304, the header will load the device driver at the address indicated by the pointer so that the device driver is loaded into the computer's memory for operation. However, an alternative method for operating the device driver permits the device driver to be executed directly from the read only memory positioned in the upper memory region of the computer's address space. In accordance with this method, the operating system passes a pointer to the header to identify where the device driver is to be stored. However, instead of storing the complete device driver in the computer's memory, the header will only store variables needed during operation of the device driver along with a small number of instructions and data to identify where the device driver is stored in the upper memory region. The complete device driver code will not be copied into the computer's memory space. Thereafter, access to the device driver will be in the upper memory region via the small driver identification code loaded in the computer's memory. This method of operation provides significant savings in the random access memory of the computer.

From the foregoing, it will be appreciated that, although specific embodiments of the invention have been described herein for purposes of illustration, various modifications may be made without deviating from the spirit and scope of the invention. Accordingly, the invention is not limited except as by the appended claims.

Claims

1. A method for loading a plurality of device drivers on a personal computer wherein the personal computer includes a computer memory, said method comprising the steps of:

(a) providing an operating system portion for locating the plurality of device drivers, the operating system portion including an operating system size identifier for identifying the size of the operating system portion and a driver signature;

(b) providing a header for each of the plurality of device drivers wherein the header includes the driver signature, a driver size identifier for device driver to identify the size of the device driver, and initialization instructions for each device driver for loading and initializing the device driver, and wherein the size of each header is known;

(c) storing the operating system portion, the headers, and the plurality of device drivers in a contiguous portion of an adapter memory wherein the operating system portion is provided first followed by the plurality of device drivers and wherein each header is stored in a memory location proximate its corresponding device driver and at an address less than that of the location of its corresponding device driver;

(d) accessing the operating system size identifier to determine the location of the end of the operating system portion and examining the location of the end of the operating system portion for the presence of the driver signature and, if the driver signature is found performing steps (e) and (f) for a current header and its corresponding current device driver and, if no driver signature is found determining that no device drivers are to be loaded;

(e) executing the initialization instructions of the current header to load the current device driver in the computer memory and to initialize the current device driver, wherein the current header is the header associated with the driver signature located in step (d) and wherein the current device driver is the device driver corresponding to the current header; and

(f) accessing the driver size identifier of the current header to determine the location of the end of the current device driver and examining the location of the end of the current device driver for the presence of the driver signature and, if a driver signature is found repeating steps (e) and (f) for a new current header and its corresponding current device driver and, if no driver signature is found determining that the last device drivers has been loaded.

2. A method for loading a plurality of device drivers into a computer memory of a personal computer wherein the computer memory includes a plurality of addresses for storing information, said method comprising the steps of:

(a) storing the plurality of device drivers in an adapter memory with a respective plurality of headers;

(b) providing each of the plurality of headers with a common signature, information for identifying another header, and instructions for loading the corresponding device driver;

(c) locating one of the plurality of headers;

(d) executing the instructions associated with the located one of the plurality of headers to load the located one of the plurality of device drivers;

(e) accessing the information for identifying another header associated with the located one of the plurality of headers;

(f) determining whether all device drivers have been loaded and, if not, locating the another header identified in step (e) and repeating steps (d)-(f).

3. The method as recited in claim 2 wherein step (c), locating one of the plurality of headers, comprises the substep of:

(g) storing information in the adapter memory for locating the one of the plurality of headers.

4. The method as recited in claim 2 wherein step (c), locating one of the plurality of headers, comprises the substep of:

(h) storing an operating system portion in the adapter memory wherein the operating system portion includes information identifying a contiguous address and wherein the contiguous address is an address that is contiguous with an address of the operating system portion;

(i) storing one of the plurality of headers so that a portion of the header is stored at the contiguous address; and

(j) accessing the operating system portion to determine the identity of the contiguous address and thereby locate one of the plurality of headers.

5. A method for loading a plurality of device drivers into a computer memory of a personal computer wherein the computer memory includes a plurality of storage locations for storing information and wherein each storage location has a storage address, said method comprising the steps of:

(a) providing each of the plurality of device drivers with a header wherein each header includes loading instructions and a driver block size identifier;

(b) storing the plurality of device drivers with their header in a corresponding plurality of driver blocks wherein each of the plurality of driver blocks is a portion of the plurality of storage locations and wherein the addresses for each of the plurality of driver blocks are sequential, each of the plurality of driver blocks including one of the plurality of device drivers and its corresponding header, each of the plurality of driver blocks being stored so that the addresses for the plurality of driver blocks are sequential;

(c) storing an operating system portion in a portion of the plurality of storage locations so that the addresses for the operating system portion and the plurality of driver blocks are sequential, the operating system portion including information for identifying the first of the plurality of driver blocks wherein the first of the plurality of driver blocks is the one of the plurality of driver blocks whose addresses are sequential with the addresses of the operating system portion;

(d) accessing the operating system portion to locate the first of the plurality of driver blocks, using the loading instructions to load the one of the plurality of device drivers stored in the first of the plurality of driver blocks, and using the driver block size identifier to determine the size of the first driver block and thereby determine the location of the next one of the plurality of driver blocks;

(e) locating a subject driver block wherein the subject driver block is the next one of the plurality of driver blocks and using the loading instructions to load the one of the plurality of device drivers stored in the subject driver block;

(f) using the driver block size identifier of the one of the plurality of headers of the subject driver block to determine the size of the subject driver block and thereby to determine the location of the next one of the plurality of driver blocks, and performing step (g); and

(g) determining whether any of the plurality of device drivers of the plurality of driver blocks have not been loaded and, if not, performing step (e).

6. The method as recited in claim 5 wherein step (f), determining whether any of the plurality of device drivers of the plurality of driver blocks have not been loaded, comprises the substeps of:

(h) providing each of the plurality of headers with a common device driver signature; and

(i) after performing step (f) determining whether the information stored in the location of the determined next one of the plurality of driver blocks

includes the common driver signature and, if so, the information includes a device driver that has not been loaded and, if not, then the last device driver has been loaded.

7. A method for loading a plurality of device drivers into a computer memory of a personal computer wherein the computer memory includes a plurality of storage locations each having a storage address, said method comprising the steps of:

(a) providing a header for each of the plurality of device drivers wherein each header includes driver size information identifying the number of storage locations needed to store its corresponding device driver and loading instructions for loading its corresponding device driver;

(b) storing the plurality of headers and device drivers so that the driver size information of a subject header identifies a prospective header address;

(c) locating a header, identifying the located header as the subject header, and performing step (d);

(d) accessing the loading instructions of the subject header to load the one of the plurality of device drivers for which the subject header was provided and performing step (e);

(e) accessing the driver size information of the subject header to identify a prospective header address; and

(f) examining the information stored at the prospective header address to determine whether it is a header and, if so, identifying the header located at the prospective header address as the subject header and repeating steps (c)-(f).

8. A method for loading a plurality of device drivers into a computer, the computer including an address space that includes a computer memory and an adapter region, the plurality of device drivers being loaded into the computer memory from the adapter region, the method comprising the steps of:

(a) providing a header for each device driver wherein each header includes instructions and data for loading its corresponding device driver and wherein each header includes a driver size identifier;

(b) storing the plurality of device drivers and headers in a portion of the adapter region so that each header occupies an address space with addresses that are in sequence with the addresses of the address space occupied by its corresponding device driver and so that the plurality of headers and device drivers occupy an address space having addresses that are sequential, the driver size identifier of each header identifying the number of addresses of the address space occupied by the header and its corresponding device driver, the plurality of device drivers and headers being further

stored so that the first address of a subject header and its driver size identifier can be combined to provide the first address of another header next following the subject header;

(c) storing a operating system portion in a portion of the adapter region having addresses that are sequential with one of the headers and identifying the one of the headers as the subject header;

(d) accessing the subject header and using the instructions and data to load its corresponding device driver;

(e) using the driver size identifier of the subject header to access another header next following the subject header, identifying the another header as the subject and repeating steps (d) and (e) until all device drivers have been loaded.

9. The method as recited in claim 8 further comprising the steps of:

(f) providing each header with a common driver signature; and

(g) after identifying the another header, determining whether the another header includes the common signature and, if not, determining that all device drivers have been loaded.

10. A method for storing a plurality of device drivers in an adapter region of an address space of a computer so that the device drivers can be loaded into a computer memory of the computer, said method comprising the steps of:

(a) providing a header for each device driver wherein each header includes instructions and data for loading its corresponding device driver and wherein each header includes a driver size identifier; and

(b) storing the plurality of device drivers and headers in a portion of the adapter region so that each header occupies an address space with addresses that are in sequence with the addresses of the address space occupied by its corresponding device driver and so that the plurality of headers and device drivers occupy an address space having addresses that are sequential, the driver size identifier of each header identifying the number of addresses of the address space occupied by the header and its corresponding device driver, the plurality of device drivers and headers being further stored so that the first address of a subject header and its driver size identifier can be combined to provide the first address of another header next following the subject header.

11. The method as recited in claim 10 further comprising the step of:

(c) storing an operating system portion in a portion of the adapter region having addresses that are sequential with one of the headers thereby to locate all of the plurality of device drivers and headers.

12. The method as recited in claim 10 further comprising the step of:

(d) providing each header with a common driver signature so that after each device driver is loaded and the another header next following the subject header is identified, the another header can be examined for the common driver signature to confirm that it is a header.

13. A method for preparing a plurality of device drivers for operation wherein each device driver is associated with a header, said method comprising the steps of:

(a) identifying a current driver scan address;

(b) accessing the current driver scan address and determining whether the information stored at the current driver scan address is associated with a header and, if so, performing steps (c)-(d), and, if not, determining that all drivers have been loaded;

(c) executing the instructions of the header associated with the current driver scan address to prepare the device driver for operation; and

(d) accessing a driver size identifier of the header and combining the driver size identifier of the header with the current driver scan address to identify the new current driver scan address and repeating step (b).

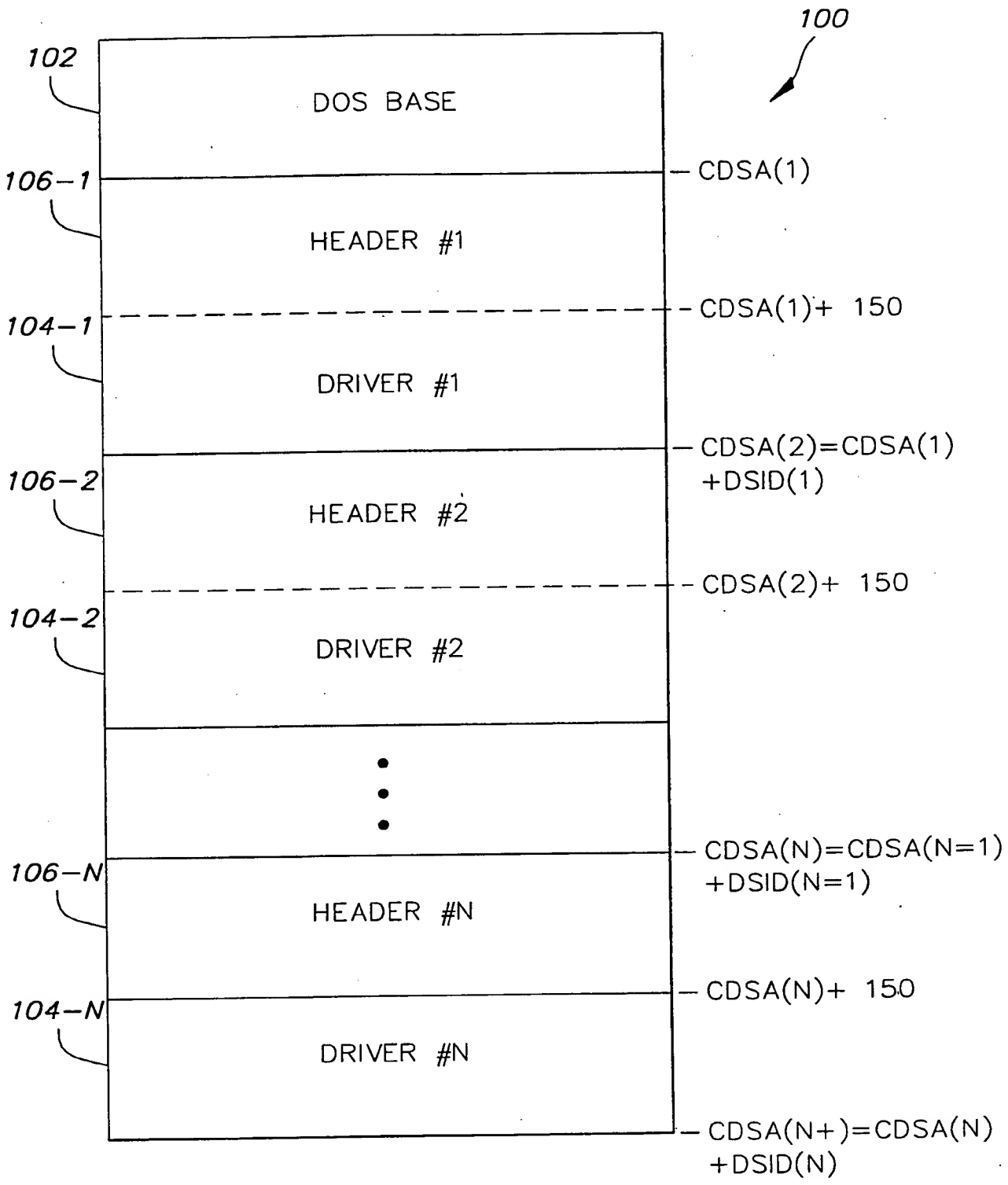

100
102
DOS BASE
CDSA(1)
106-1
HEADER #1
CDSA(1)+ 150
104-1
DRIVER #1
CDSA(2)=CDSA(1)
+DSID(1)
106-2
HEADER #2
CDSA(2)+ 150
104-2
DRIVER #2
CDSA(N)=CDSA(N=1)
+DSID(N=1)
106-N
HEADER #N
CDSA(N)+ 150
104-N
DRIVER #N
CDSA(N+)=CDSA(N)
+DSID(N)

Figure 1

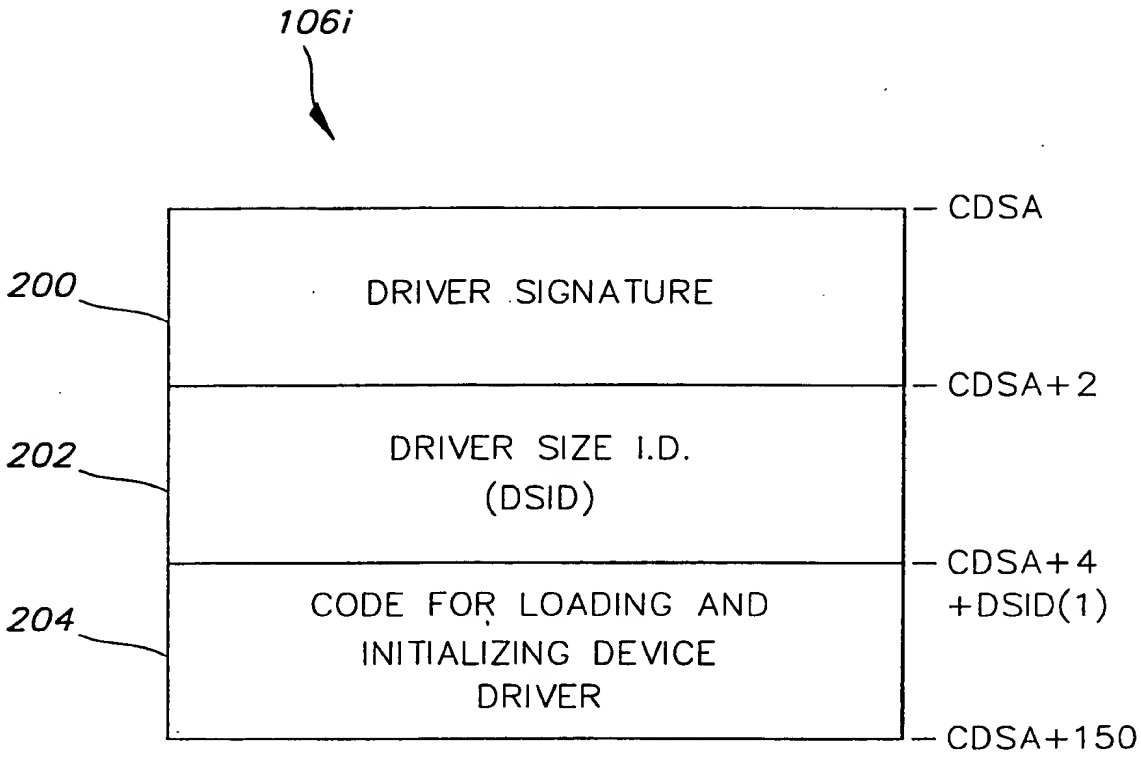

106i
200
DRIVER SIGNATURE
CDSA
CDSA+2
202
DRIVER SIZE I.D.
(DSID)
CDSA+4
+DSID(1)
204
CODE FOR LOADING AND
INITIALIZING DEVICE
DRIVER
CDSA+150

Figure 2

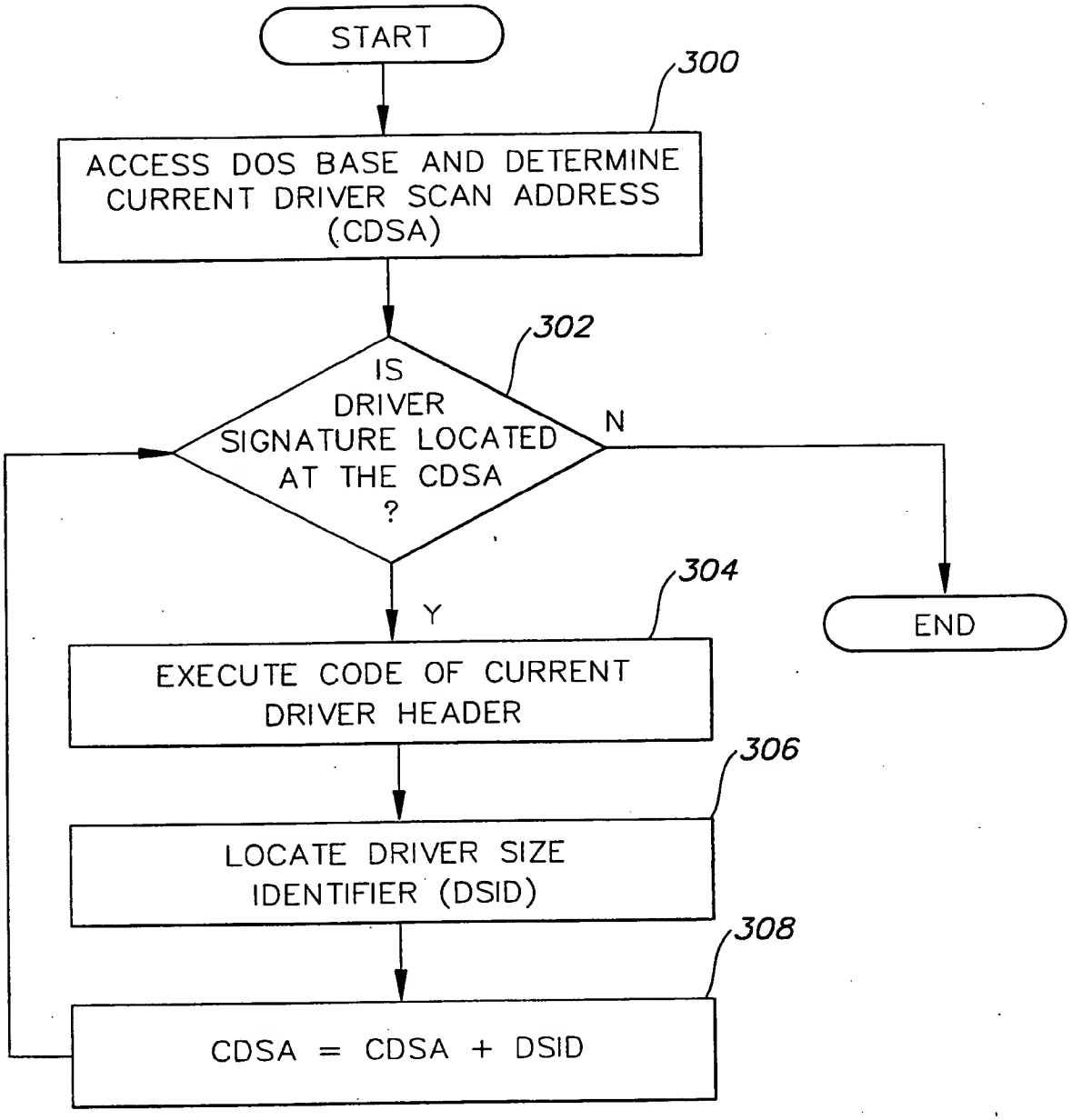

START
300
ACCESS DOS BASE AND DETERMINE CURRENT DRIVER SCAN ADDRESS (CDSA)
302
IS DRIVER SIGNATURE LOCATED AT THE CDSA ?
N
Y
END
304
EXECUTE CODE OF CURRENT DRIVER HEADER
306
LOCATE DRIVER SIZE IDENTIFIER (DSID)
308
CDSA = CDSA + DSID

Figure 3

INTERNATIONAL SEARCH REPORT

Int. ional Application No
PCT/US 93/11227

A. CLASSIFICATION OF SUBJECT MATTER
IPC 5 G06F9/445

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 5 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	PC TECH JOURNAL, vol. 3, no. 5 , May 1985 , US, pages 76 - 95 STAN MITCHELL: 'Building Device Drivers' see page 79, left column, line 9 - line 58; figure 1; tables 1,2 see page 80, middle column, line 7 - page 81, right column, line 9 ---	1-13
Y	GB,A,2 203 869 (APPLE COMPUTER INC.) 26 October 1988 see abstract; figure 3 see page 1, line 27 - page 3, line 6; figure 1 see page 4 - page 5 see page 6, line 1 - page 9, line 26 see page 13, line 27 - page 17, line 4; figures 7,8 -----	1-13

☐ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
7 March 1994	25. 03. 94
Name and mailing address of the ISA European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+ 31-70) 340-2040, Tx. 31 651 epo nl, Fax (+ 31-70) 340-3016	Authorized officer Wiltink, J

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Information on patent family members

Int ional Application No
PCT/US 93/11227

Patent document cited in search report	Publication date	Patent family member(s)		Publication date
GB-A-2203869	26-10-88	AU-B-	611137	06-06-91
		AU-A-	1418588	20-10-88
		CA-A-	1296806	03-03-92
		DE-A-	3812607	27-10-88
		FR-A-	2614122	21-10-88
		JP-A-	63279340	16-11-88